UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/792,122	03/03/2004	Steven T. Antoch	003797.00767	3925

67321 7590 01/21/2009
BIRCH, STEWART, KOLASCH & BIRCH, LLP
PO Box 747
FALLS CHURCH, VA 22040-0747

EXAMINER	
GOFMAN, ALEX N	

ART UNIT	PAPER NUMBER
2162	

MAIL DATE	DELIVERY MODE
01/21/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/792,122	ANTOCH, STEVEN T.	
	Examiner	Art Unit	
	ALEX GOFMAN	2162	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 November 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4,7-10,12,14,16,18,20 and 21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,4,7-10,12,14,16,18,20 and 21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *03 March 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on November 10, 2008 has been entered.

REMARKS

Claims 16 and 18 recite "A method implemented at least in part by a *computing device*..." (emphasis added). The *computing device* is interpreted as a hardware computer 100 as illustrated in Figure 1.

Response to Arguments

2. Applicant's arguments with respect to claims 1, 4, 7-10, 12, 14, 16, 18, and 20-21 have been considered but are moot in view of the new ground(s) of rejection using previously cited prior art. Responses to specific arguments of newly added limitations are addressed in the body of the rejection.

Claim Objections

3. **Claims 1, 9, 16 and 18** are objected to because of the following informalities: The Claims recite elements such as "a base *filed* handler class" and "*filed* handler object" (emphasis added). The Examiner assumes that the word "filed" should in fact be "field." If the word should in fact be "filed," it would render the claim as indefinite because the Examiner cannot apply the term "filed" in light of the specification of the instant application. There may be other instances of such typos throughout which should also be corrected. Appropriate correction is required.

Claim 8 is objected to because it states "wherein the data structure further comprises ***d)*** a meta-class…" (emphasis added). However, Claim 1 already has a limitation ***d)*** (emphasis added).

Claim Rejections - 35 USC § 112

4. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

5. **Claims 1, 9, 16 and 18** rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the

time the application was filed, had possession of the claimed invention. The Claims describe concepts of type safe and non-type safe values and access to such values. However, the specification of the instant application neither uses nor describes the application of type safety to the overall invention.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1, 4, 7-10, 12, 14, 16, 18, and 20-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wall et al (US Patent Application Publication 2002/0087557), hereinafter, Wall in view of Matula et al (US Patent Application Publication 2002/0165995), hereinafter, Matula and further in view of Coad et al (Patent 6,851,105), hereinafter Coad.

Claim 1: Wall discloses a computer-readable storage medium having stored thereon a data structure, the data structure separating storage of an attribute value from handling of the attribute value, comprising:

a. a model element class configured to implement the constructs described by metadata; the model element class storing an attribute value in a private member field of the model element class in the same memory block as declaring class [0064]. [The "constructs described by metadata" is a class declaration of how each attribute function, i.e. integer, string, etc. As for the "same memory block," there is no explicit definition as to what a memory block is in the specification of the instant specification, and thus a memory block could be a hard drive, a database, etc.]

b. Wall discloses wherein a handler class has public access to an enclosing element's private members stored in the private member field of the model element class [0065], but does not explicitly disclose a nested handler class, wherein the nested handler class is a subclass of a generic handler class and inherits base functionality from the generic handler class. However, Matula does [0051]. A nested handler class, a subclass in Matula, is created in order to generate a particular instance of an interface. Once the instance is created, the value retrieved by the subclass is stored in the main class.

It would have been obvious for one of ordinary skill in the art at the time the invention was made to declare a nested handler class, the nested handler class being a subclass of a generic handler class and inherits base functionality from the generic handler class. One would have been motivated to do so in order to store a value requested in the main requesting class.

d. Wall as modified discloses a model element field handler object for handling inlined field values of the model element class configured to access the attribute value stored in the model element class, wherein model element field handler object comprises a typed model element field handler subclass defining a get value function wherein the get value function is configured to access the model element class and return the attribute value directly upon request [Wall 0065 and Matula [0051]. [The claim describes a method for defining a "get value function" and then returning the value requested from the "get function." As for the "inlined field values," the specification of the instant application only describes the inlined values as "located in the same memory block as the declaring class." The discussion for "same memory block" is found above.]

It would have been obvious for one of ordinary skill in the art at the time the invention was made to use a singleton pattern as an abstract class. One would have been motivated to do so in order to have only one instance of a class, and thereby using only the single object to coordinate actions across a system.

e. a base field handler class which acts as an intermediary generic mechanism when getting the field values in the model element field handler object, wherein the get value function is configured to:

i) dispatch to a sub-classed get field value function in the first subclass which is type safe [0064-0065].

ii) access the model element class and return the attribute value directly upon request [0064-0065].

iii) provide an entry point for non-type safe application programming interface so that general purpose client code can access the field values without relying on type safety [0044, 0054]. [Please see the 35 USC 112 1st paragraph rejection above. The API provides access to a legacy system which includes both type safe and non-type safe values.]

wherein the storage of the attribute value is separate from handling of the attribute value [0031]. [Wall discloses the use of encapsulation. It is used for classes and its subclasses. For example, when the "handler" object (as in [0040 and 0065]) needs to get an attribute from an object of a subclass or a private class, the attribute is processed in the superclass. Thus, the storage of an attribute is separate from the handling of the attribute.]

Claim 4: Wall as modified discloses the medium of Claim 1 above, and further discloses wherein the model element field handler object sets the attribute value sorted in the model element class [0037].

Claim 7: Wall as modified discloses the medium of Claim 1 above, and further discloses wherein the typed model element field handler subclass defines a set value function for setting the attribute value [0037].

Claim 8: Wall as modified discloses the medium of Claim 1 above, and further discloses wherein the data structure further comprises a meta-class information object for storing data associated with the model element [0064].

Claim 9: Wall discloses a computer-readable storage medium having stored thereon a data structure, the data structure separating storage of an attribute value from handling of the attribute value, comprising:

a. a container for storing recta-data in a tree structure [0062].

b. a model element class configured to implement the constructs described by metadata; the model element class storing an attribute value in a private member field of the model element class in the same memory block as a declaring class [0064]. [The "constructs described by metadata" is a class declaration of how each attribute function, i.e. integer, string, etc. As for the "same memory block," there is no explicit definition as to what a memory block is in the specification of the instant specification, and thus a memory block could be a hard drive, a database, etc.]

c. Wall discloses wherein a handler class has public access to an enclosing element's private members stored in the private member field of the model element class [0065], but does not explicitly disclose a nested handler class, wherein the nested handler class is a subclass of a generic handler class and inherits base functionality from the generic handler class. However, Matula does [0051]. A nested handler class, a subclass in Matula, is created in order to

generate a particular instance of an interface. Once the instance is created, the value retrieved by the subclass is stored in the main class.

It would have been obvious for one of ordinary skill in the art at the time the invention was made to declare a nested handler class, the nested handler class being a subclass of a generic handler class and inherits base functionality from the generic handler class. One would have been motivated to do so in order to store a value requested in the main requesting class.

d. a meta-class information object configured to store data associated with the model element [0064].

e. a meta-attribute information object configured to describe attributes of the model element class [0064].

f. Wall as modified discloses a model element field handler object for handling inlined field values of the model element class configured to access the attribute value stored in the model element class, wherein model element field handler object comprises a typed model element field handler subclass defining a set value function wherein the setting value function [Wall 0037 and Matula [0051]]. [As for the "inlined field values," the specification of the instant application only describes the inlined values as "located in the same memory block as the declaring class." The discussion for "same memory block" is found above.]

However, Wall does not explicitly disclose that the model element field handler object comprises *a singleton pattern as an abstract base class*. However, Coad discloses using a singleton pattern (Col 8 ln 29-47, Col 23 ln 5-23). A singleton pattern, according to Coad, is a class with only one instance and contains only provides a global point of access to it. Also, a singleton pattern is performed through an interface class, which is an abstract class.

It would have been obvious for one of ordinary skill in the art at the time the invention was made to use a singleton pattern as an abstract class. One would have been motivated to do so in order to have only one instance of a class, and thereby using only the single object to coordinate actions across a system.

g. a base field handler class which acts as an intermediary generic mechanism when setting the field values in the model element field handler object, wherein the set value function is configured to:

i) provide validation of a new value [0036]

ii) Wall does explicitly disclose recording necessary undo information. However the background section of the instant application (Antoch) does [04]. Antoch discloses having an undo operation in a meta-model. While the claim language only requires that the undo information is only recorded, it would have been obvious for one of ordinary skill in the art at the time the invention was made to record undo information if an application supports an undo operation. One would have been motivated to do so in order complete an undo operation.

iii) dispatch to a sub-classed set field value function in the first subclass which is type safe [0064-0065].

h. Wall discloses wherein a handler class has public access to an enclosing element's private members stored in the private member field of the model element class [0065], but does not explicitly disclose wherein the nested handler class is configured to directly access data in the model element class as the nested handler class has public access to the private members of the model element class. However, Matula does [0051]. The nested handler class would have all the functionality of the actual handler class since the subclass inherits all functionality of its superclass. The nested handler class would be able to access the private class based on permissions given to that specific instance as described in Wall [0037, 0066].

Thus, it would have been obvious for one of ordinary skill in the art at the time the invention was made to have the nested handler class is configured to directly access data in the model element class as the nested handler class has public access to the private members of the model element class. One would have been motivated to do so in order have specific functionality of a class by setting various permissions.

i. Wall as modified further discloses wherein the storage of the attribute value is separate from handling of the attribute value [0031]. [Wall discloses the use of encapsulation. It is used for classes and its subclasses. For example,

when the "handler" object (as in [0040 and 0065]) needs to get an attribute from an object of a subclass or a private class, the attribute is processed in the superclass. Thus, the storage of an attribute is separate from the handling of the attribute.

Claim 10: Wall as modified discloses the medium of Claim 9 above, and further discloses wherein the container comprises a store acting as a root of the tree structure [Fig. 5].

Claim 12: Wall as modified discloses the medium of Claim 9 above, and further discloses wherein the model element field handler object sets the attribute value stored in the model element class [0040].

Claim 14: Wall as modified discloses the medium of Claim 9 above, and further discloses wherein the typed model element field handler subclass defines a get value function for accessing the attribute value [0040].

Claim 16: Wall discloses a method implemented at least in part by a computing device, the computing device accessing an attribute value within a data structure, the data structure separating storage of the attribute value from handling of the attribute value, the method comprising:

a. storing the attribute value in a private member field of a model element class in a same memory block as a declaring class [0064-0065]. [As for the "same memory block," there is no explicit definition as to what a memory block is

in the specification of the instant specification, and thus a memory block could be a hard drive, a database, etc.]

b. Wall discloses wherein a handler class has public access to an enclosing element's private members stored in the private member field of the model element class [0065], but does not explicitly disclose declaring a nested handler class, the nested handler class being a subclass of a generic handler class and inherits base functionality from the generic handler class. However, Matula does [0051]. A nested handler class, a subclass in Matula, is created in order to generate a particular instance of an interface. Once the instance is created, the value retrieved by the subclass is stored in the main class.

It would have been obvious for one of ordinary skill in the art at the time the invention was made to declare a nested handler class, the nested handler class being a subclass of a generic handler class and inherits base functionality from the generic handler class. One would have been motivated to do so in order to store a value requested in the main requesting class.

c. Wall as modified discloses a model element field handler object for handling inlined field values of the model element class configured to access the attribute value stored in the model element class, wherein model element field handler object comprises a typed model element field handler subclass defining a get value function wherein the get value function is configured to access the model element class and return the attribute value directly upon request [Wall

0065 and Matula [0051]]. [The claim describes a method for defining a "get value function" and then returning the value requested from the "get function." As for the "inlined field values," the specification of the instant application only describes the inlined values as "located in the same memory block as the declaring class." The discussion for "same memory block" is found above.]

However, Wall does not explicitly disclose that the model element field handler object comprises *a singleton pattern as an abstract base class*. However, Coad discloses using a singleton pattern (Col 8 ln 29-47, Col 23 ln 5-23). A singleton pattern, according to Coad, is a class with only one instance and contains only provides a global point of access to it. Also, a singleton pattern is performed through an interface class, which is an abstract class.

It would have been obvious for one of ordinary skill in the art at the time the invention was made to use a singleton pattern as an abstract class. One would have been motivated to do so in order to have only one instance of a class, and thereby using only the single object to coordinate actions across a system.

d. issuing the get value function to obtain the attribute value from the model element class [0040, 0064].

e. receiving the attribute value from the model element class [0065].

f. providing a base field handler class which acts as an intermediary generic mechanism when getting the field values in the model element field handler object, wherein the get value function is configured to:

i) dispatch to a sub-classed get field value function in the first subclass which is type safe [0064-0065].

ii) access the model element class and return the attribute value directly upon request [0064-0065].

iii) provide an entry point for non-type safe application programming interface so that general purpose client code can access the field values without relying on type safety [0044, 0054]. [Please see the 35 USC 112 1st paragraph rejection above. The API provides access to a legacy system which includes both type safe and non-type safe values.]

g. Wall as modified further discloses wherein the storage of the attribute value is separate from handling of the attribute value [0031]. [Wall discloses the use of encapsulation. It is used for classes and its subclasses. For example, when the "handler" object (as in [0040 and 0065]) needs to get an attribute from an object of a subclass or a private class, the attribute is processed in the superclass. Thus, the storage of an attribute is separate from the handling of the attribute.

Claim 18: Wall discloses a method implemented at least in part by a computing device, the computing device setting an attribute value within a data structure, the data structure separating storage of the attribute value from handling of the attribute value, the method comprising:

a. Wall discloses wherein a handler class has public access to an enclosing element's private members stored in the private member field of the model element class [0065], but does not explicitly disclose declaring a nested handler class, the nested handler class being a subclass of a generic handler class and inherits base functionality from the generic handler class. However, Matula does [0051]. A nested handler class, a subclass in Matula, is created in order to generate a particular instance of an interface. Once the instance is created, the value retrieved by the subclass is stored in the main class.

It would have been obvious for one of ordinary skill in the art at the time the invention was made to declare a nested handler class, the nested handler class being a subclass of a generic handler class and inherits base functionality from the generic handler class. One would have been motivated to do so in order to store a value requested in the main requesting class.

b. Wall as modified discloses a model element field handler object for handling inlined field values of the model element class configured to access the attribute value stored in the model element class, wherein model element field handler object comprises a typed model element field handler subclass defining a set value function wherein the setting value function [Wall 0037 and Matula [0051]]. [As for the "inlined field values," the specification of the instant application only describes the inlined values as "located in the same memory

block as the declaring class." The discussion for "same memory block" is found above.]

However, Wall does not explicitly disclose that the model element field handler object comprises *a singleton pattern as an abstract base class*. However, Coad discloses using a singleton pattern (Col 8 ln 29-47, Col 23 ln 5-23). A singleton pattern, according to Coad, is a class with only one instance and contains only provides a global point of access to it. Also, a singleton pattern is performed through an interface class, which is an abstract class.

It would have been obvious for one of ordinary skill in the art at the time the invention was made to use a singleton pattern as an abstract class. One would have been motivated to do so in order to have only one instance of a class, and thereby using only the single object to coordinate actions across a system.

c. issuing the set value function to set the attribute value for the model element class [0040, 0064].

d. setting the attribute value [0040, 0064].

e. storing the attribute value in a private member field of the model element class in a same memory block as a declaring class [0064]. [The "constructs described by metadata" is a class declaration of how each attribute function, i.e. integer, string, etc. As for the "same memory block," there is no explicit definition as to what a memory block is in the specification of the instant specification, and thus a memory block could be a hard drive, a database, etc.]

f. providing a base field handler class which acts as an intermediary generic mechanism when setting the field values in the model element field handler object, wherein the set value function is configured to:

i) provide validation of a new value [0036].

ii) Wall does explicitly disclose recording necessary undo information. However the background section of the instant application (Antoch) does [04]. Antoch discloses having an undo operation in a meta-model. While the claim language only requires that the undo information is only recorded, it would have been obvious for one of ordinary skill in the art at the time the invention was made to record undo information if an application supports an undo operation. One would have been motivated to do so in order complete an undo operation.

iii) dispatch to a sub-classed set field value function in the first subclass which is type safe [0064-0065].

g. Wall discloses wherein a handler class has public access to an enclosing element's private members stored in the private member field of the model element class [0065], but does not explicitly disclose wherein the nested handler class is configured to directly access data in the model element class as the nested handler class has public access to the private members of the model element class. However, Matula does [0051]. The nested handler class would have all the functionality of the actual handler class since the subclass inherits all functionality of its superclass. The nested handler class would be able to access

the private class based on permissions given to that specific instance as described in Wall [0037, 0066].

Thus, it would have been obvious for one of ordinary skill in the art at the time the invention was made to have the nested handler class is configured to directly access data in the model element class as the nested handler class has public access to the private members of the model element class. One would have been motivated to do so in order have specific functionality of a class by setting various permissions.

h. Wall as modified further discloses wherein the storage of the attribute value is separate from handling of the attribute value [0031]. [Wall discloses the use of encapsulation. It is used for classes and its subclasses. For example, when the "handler" object (as in [0040 and 0065]) needs to get an attribute from an object of a subclass or a private class, the attribute is processed in the superclass. Thus, the storage of an attribute is separate from the handling of the attribute.

Claims 20 and 21: Wall as modified discloses the medium of Claims 1 and 9 above, but Wall does not explicitly disclose wherein the singleton pattern enables the data structure to instantiate only one instance of a particular object which is used for supplying functionality for other users who wish to call that one instance. However, Coad does (Col 8 ln 29-47, Col 23 ln 5-23). A singleton pattern, according to Coad, is a class with only one instance and contains only provides a global

point of access to it. Also, a singleton pattern is performed through an interface class, which is an abstract class.

It would have been obvious for one of ordinary skill in the art at the time the invention was made to have the singleton pattern enables the data structure to instantiate only one instance of a particular object which is used for supplying functionality for other users who wish to call that one instance. One would have been motivated to do so in order to have only one instance of a class, and thereby using only the single object to coordinate actions across a system.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to ALEX GOFMAN whose telephone number is (571)270-1072. The examiner can normally be reached on Mon-Fri 9am-3pm EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Breene can be reached on (571)272-4107. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Alex Gofman
Examiner
Art Unit 2162

AG
1-12-09

/John Breene/
Supervisory Patent Examiner, Art Unit 2162